Derivative-Free Multi-Agent Optimization

Jeffrey Larson

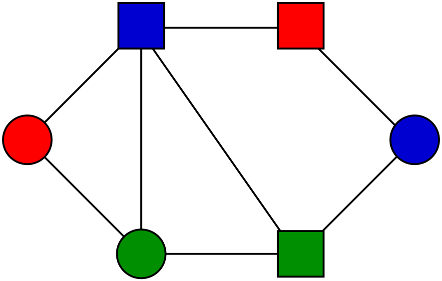
Argonne National Laboratory

May 19, 2014

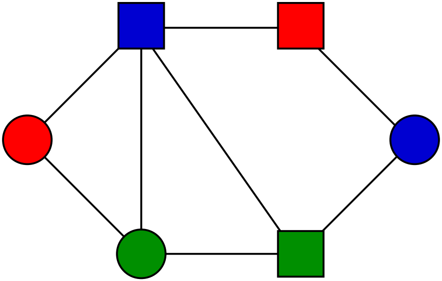
Contents

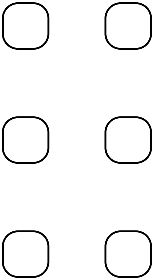
Motivation

Motivation

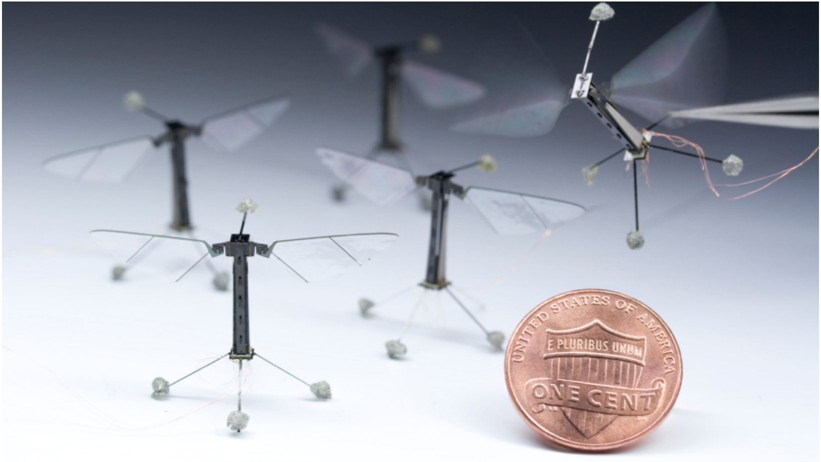


Motivation





Motivation



Credit: *RoboBees Project, Harvard University*

Problem Statement

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \mathbb{E}\left[\sum_{i=1}^{N} \bar{f}_i(x)\right] \\
\text{subject to} \quad & Ax \leq b \\
& x \in X
\end{aligned}
$$

- Each agent has objective $f_i(x)$ which can only be observed with additive noise $\bar{f}_i(x) = f_i(x) + \epsilon$
- ϵ has zero mean and finite variance
- X is an is a nonempty, closed, convex subset of $\mathbb{R}^n$

Outline

Goal

Agents connected by a network cooperatively minimize the global objective though they only have knowledge of their individual objectives (and shared information from the network).

- Aim: Distributed Multi-agent Derivative-free Optimization
  - At iteration j, agent i builds a model m_j^i using observed values of $\bar{f}_i$.
  - Communicate where they are going to their neighbors in the network.
  - Take the information from their neighbors for iteration $j+1$.

Outline

Goal

Agents connected by a network cooperatively minimize the global objective though they only have knowledge of their individual objectives (and shared information from the network).

- Aim: Distributed Multi-agent Derivative-free Optimization
  - At iteration j, agent i builds a model m_j^i using observed values of $\bar{f}_i$.
  - Communicate where they are going to their neighbors in the network.
  - Take the information from their neighbors for iteration $j+1$.

- Today: Distributed Multi-agent Optimization with Inexact Subproblems

Outline

Goal

Agents connected by a network cooperatively minimize the global objective though they only have knowledge of their individual objectives (and shared information from the network).

- Aim: Distributed Multi-agent Derivative-free Optimization
  - At iteration j, agent i builds a model m_j^i using observed values of $\bar{f}_i$.
  - Communicate where they are going to their neighbors in the network.
  - Take the information from their neighbors for iteration $j+1$.
- Today: Distributed Multi-agent Optimization with Inexact Subproblems
- First: Single agent case

The Problem

We want to solve:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

when $\nabla f(x)$ is unavailable and we only have access to noise-corrupted function evaluations $\bar{f}(x)$.

The Problem

We want to solve:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

when $\nabla f(x)$ is unavailable and we only have access to noise-corrupted function evaluations $\bar{f}(x)$.

Such noise may be deterministic (e.g., from iterative methods) or stochastic (e.g., from a Monte-Carlo process).

The Problem

We want to solve:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

when $\nabla f(x)$ is unavailable and we only have access to noise-corrupted function evaluations $\bar{f}(x)$.

Such noise may be deterministic (e.g., from iterative methods) or stochastic (e.g., from a Monte-Carlo process).

Model-based methods are one of the most popular methods when ∇f is unavailable, and the only recourse when noise is deterministic.

The Problem

We analyze the convergence of our method in the stochastic case:

$$\bar{f}(x) = f(x) + \epsilon,$$

where ϵ is identically distributed with mean 0 and variance $\sigma^2 < \infty$.

The Problem

We analyze the convergence of our method in the stochastic case:

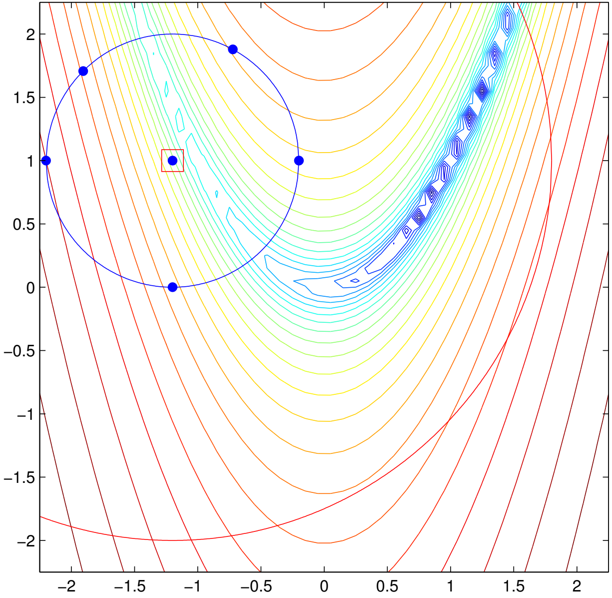
$$\bar{f}(x) = f(x) + \epsilon,$$

where ϵ is identically distributed with mean 0 and variance $\sigma^2 < \infty$.

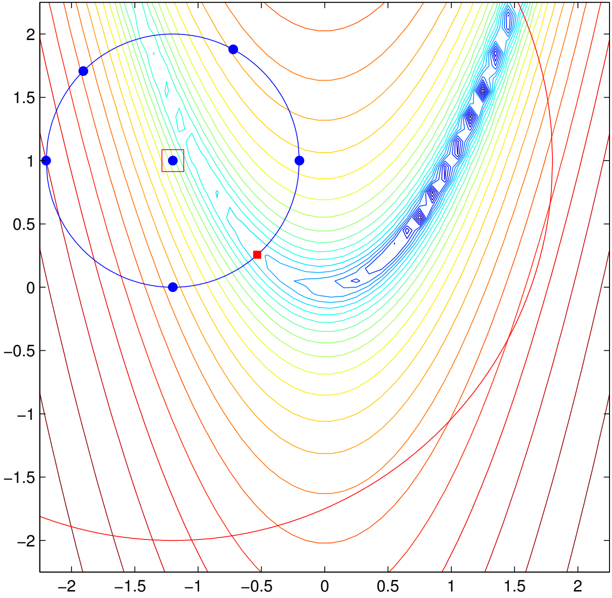
This is equivalent to solving:

$$\underset{x}{\text{minimize}}\ \mathbb{E}\left[\bar{f}(x)\right].$$

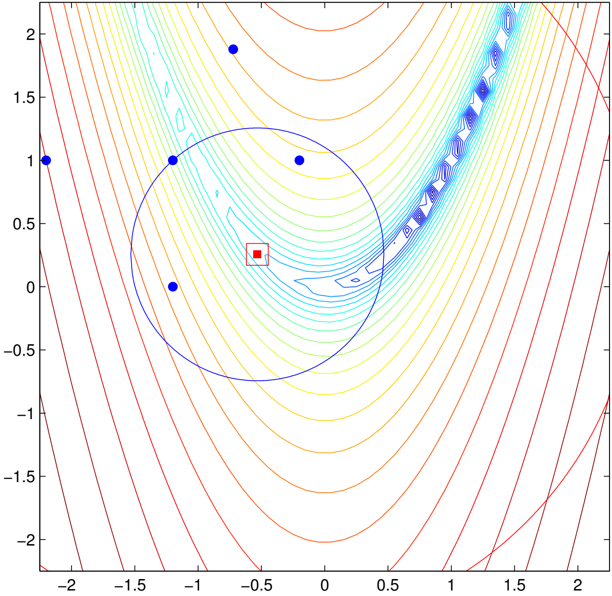
Prototype



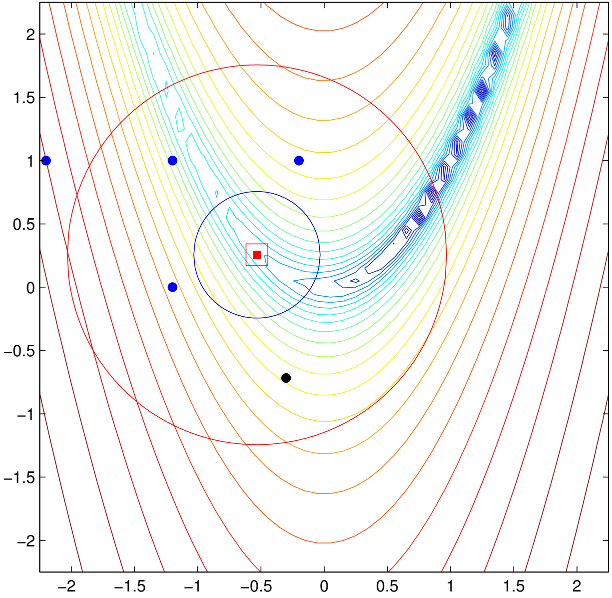
Prototype



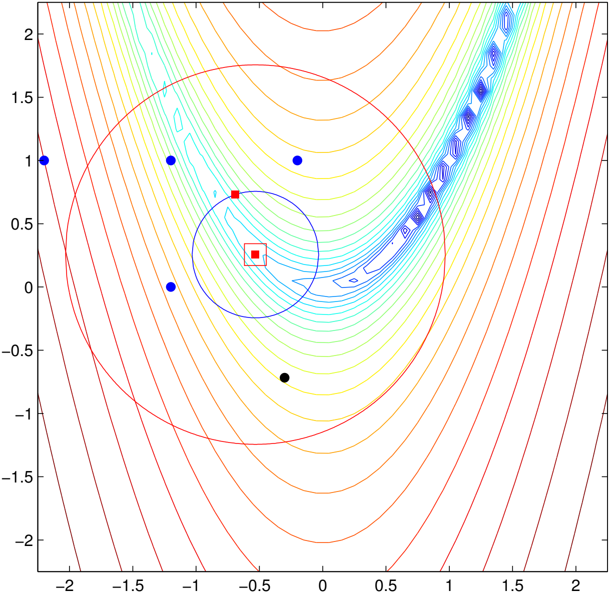
Prototype



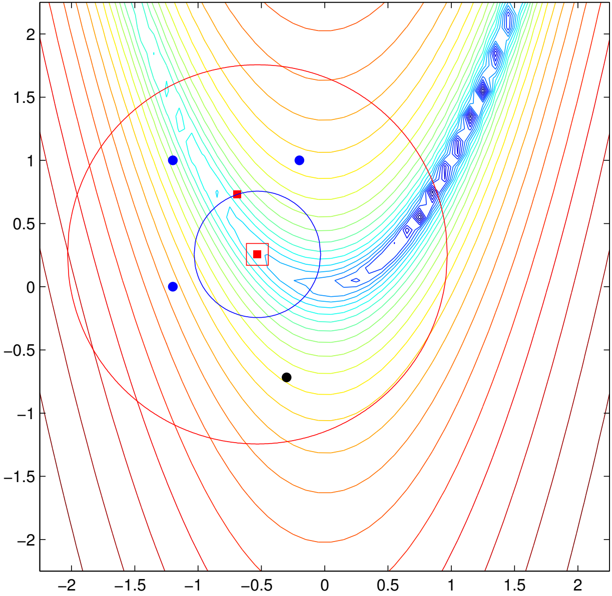
Prototype



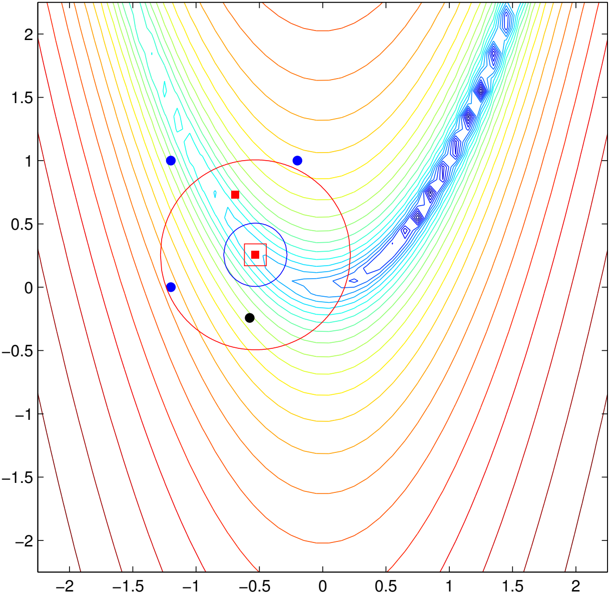
Prototype



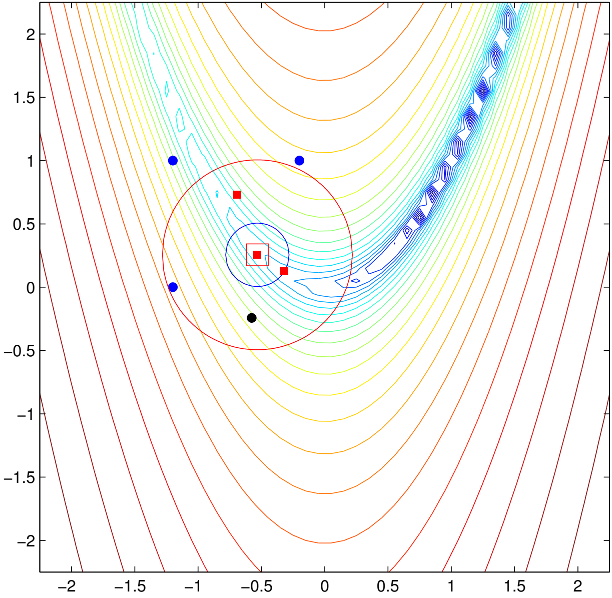
Prototype



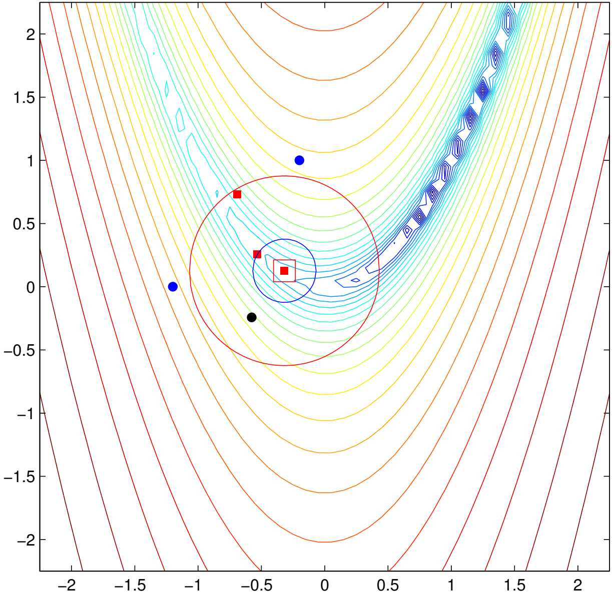
Prototype



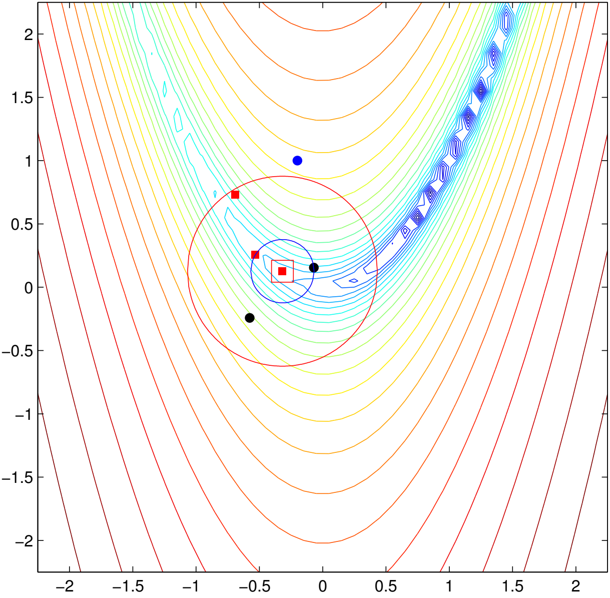
Prototype



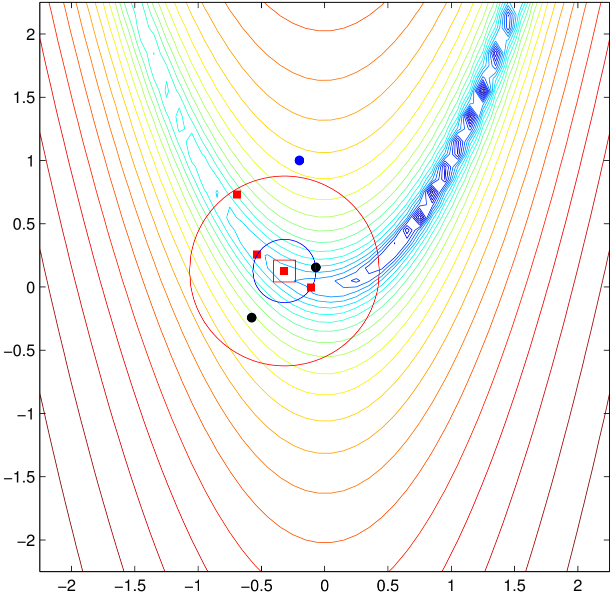
Prototype



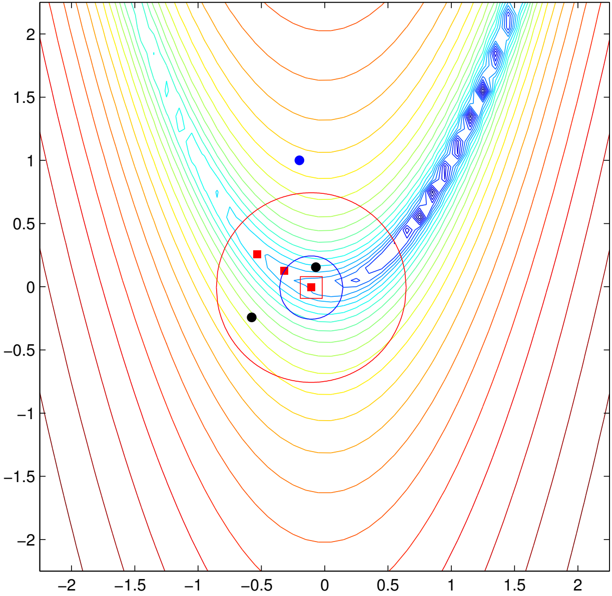
Prototype



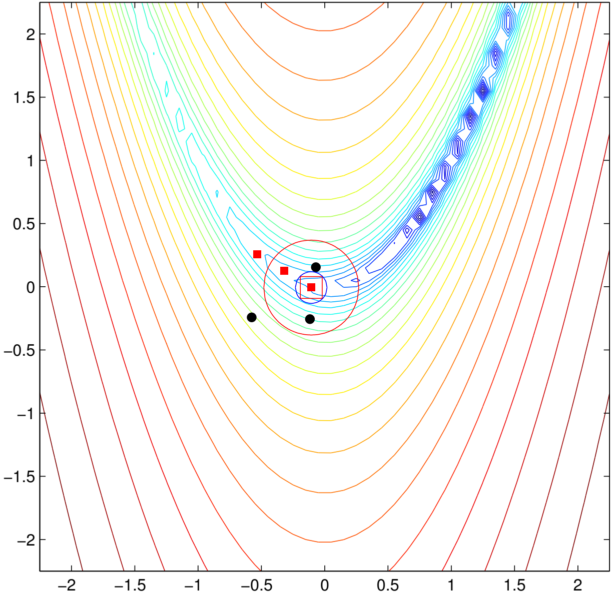
Prototype



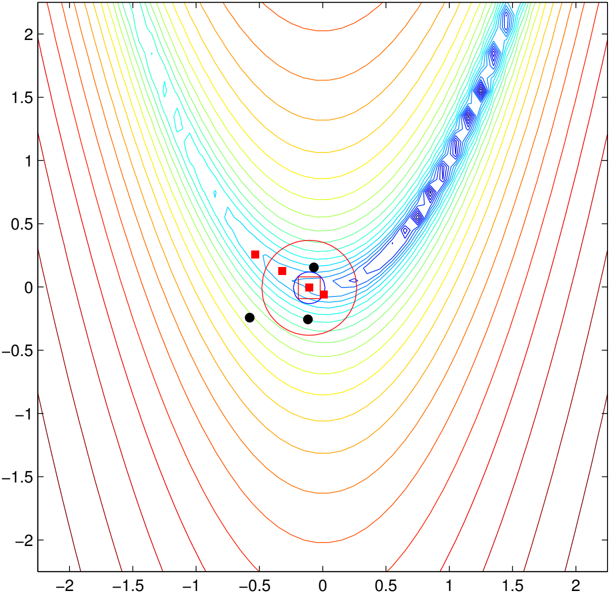
Prototype



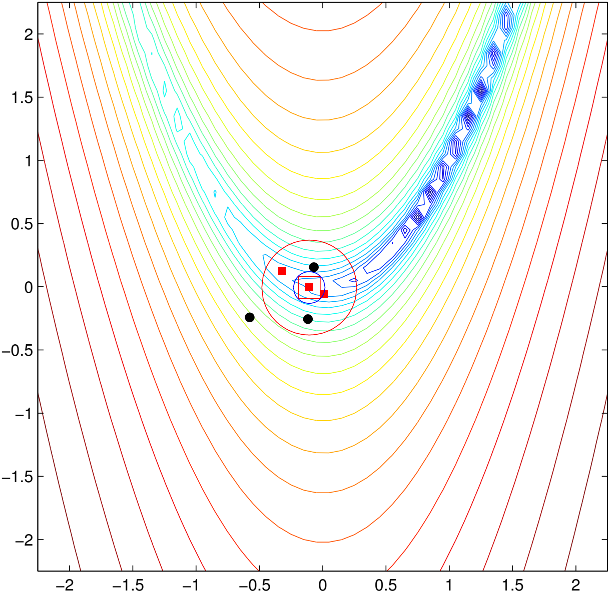
Prototype



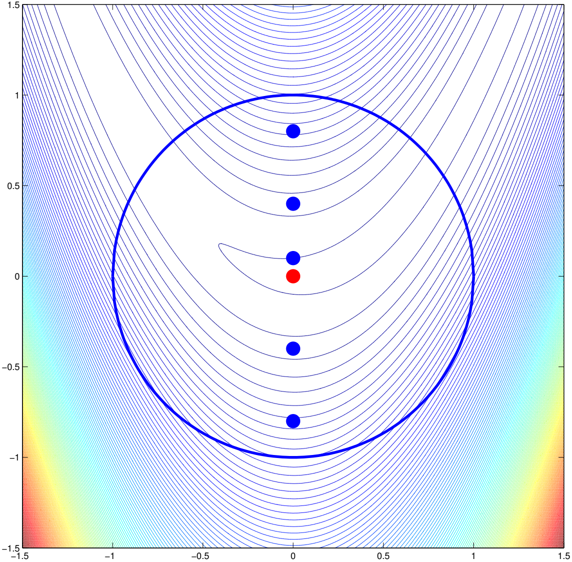
Prototype



Prototype



Strongly Λ-poised Sets



Other Approaches

- Stochastic Approximation

$$x^{k+1} = x^k + a_k G(x^k)$$

Other Approaches

- Stochastic Approximation

$$x^{k+1} = x^k + a_k G(x^k)$$

- Response Surface Methodology
  - Build models using a fixed pattern of points, for example, cubic, spherical, or orthogonal designs among many others.

Other Approaches

- Stochastic Approximation

$$x^{k+1} = x^k + a_k G(x^k)$$

- Response Surface Methodology
  - Build models using a fixed pattern of points, for example, cubic, spherical, or orthogonal designs among many others.
- Repeated Sampling
  - Take a favorite method and repeatedly evaluate the function at points of interest.

Other Approaches

- Stochastic Approximation

$$x^{k+1} = x^k + a_k G(x^k)$$

- Response Surface Methodology
  - Build models using a fixed pattern of points, for example, cubic, spherical, or orthogonal designs among many others.
- Repeated Sampling
  - Take a favorite method and repeatedly evaluate the function at points of interest.

- How to define a_k?
- What pattern creates models that best fit the function?
- Repeated evalutations provide information about the noise ϵ, not f.

Overview

We therefore desire a method that

1. Adjusts the step size as it progresses
2. Does not use a fixed design of points
3. Does not repeatedly sample points

Overview

We therefore desire a method that

1. Adjusts the step size as it progresses
2. Does not use a fixed design of points
3. Does not repeatedly sample points

We'd like the class of possible models to be general.

κ-fully Linear model

Definition

If $f \in LC$ and $\exists$ a vector $\kappa = (\kappa_{ef}, \kappa_{eg})$ of positive constants such that

- the error between the gradient of the model and the gradient of the function satisfies

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\Delta \quad \forall y \in B(x;\Delta),$$

- the error between the model and the function satisfies

$$|f(y) - m(y)| \leq \kappa_{ef}\Delta^2 \quad \forall y \in B(x;\Delta),$$

we say the model is κ-*fully linear* on $B(x;\Delta)$.

α-probabilistically κ-fully Linear model

Definition

Let $\kappa = (\kappa_{ef}, \kappa_{eg})$ be a given vector of constants, and let $\alpha \in (0, 1)$. Let $B \subset \mathbb{R}^n$ be given. A random model m_k generated at the kth iteration of an algorithm is α-probabilistically κ-fully linear on B if

$$P\left(m_k \text{ is a } \kappa\text{-fully linear model of } f \text{ on } B \middle| \mathcal{F}_{k-1}\right) \geq \alpha,$$

where $\mathcal{F}_{k-1}$ denotes the realizations of all the random events for the first $k-1$ iterations.

Regression Models can be α-probabilistically κ-fully Linear

Theorem

For a given $x \in \mathbb{R}^n$, $\Delta > 0$, $\alpha \in (0,1)$,

- $Y \subset B(x;\Delta)$ *is strongly* Λ*-poised,*
- *The noise present in* $\bar{f}$ *is i.i.d. with mean 0, variance* $\sigma^2 < \infty$,
- $|Y| \geq C/\Delta^4$,

Then there exist constants $\kappa = (\kappa_{ef}, \kappa_{eg})$ *(independent of* Δ *and* Y*) such that the linear model* m *regressing* Y *is* α*-probabilistically* κ*-fully linear on* $B(x;\Delta)$.

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{\bar{f}(x^k) - \bar{f}(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{m_k(x^k) - m_k(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{m_k(x^k) - \hat{m}_k(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

Measuring Progress

In traditional trust region methods, if $x^k + s^k$ is the minimizer of m_k, the success of moving from x^k to $x^k + s^k$ is measured by

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m_k(x^k) - m_k(x^k + s^k)}$$

In the stochastic case, a similar calculation is not obvious.

$$\rho_k = \frac{F_k^0 - F_k^s}{m_k(x^k) - m_k(x^k + s^k)}$$

Algorithm 1: A trust region algorithm to minimize a stochastic function

Set $k = 0$;

Start

Build a α-probabilistically κ-fully linear model m_k on $B(x^k; \Delta_k)$;

Compute $s^k = \arg \min_{s:\|x^k - s\| \leq \Delta_k} m_k(s)$;

if $m_k(s^k) - m_k(x^k + s^k) \geq \beta\Delta_k$ **then**

    Calculate $\rho_k = \dfrac{F_k^0 - F_k^s}{m_k(x^k) - m_k(x^k + s^k)}$;

    if $\rho_k \geq \eta$ **then**

        Calculate $x^{k+1} = x^k + s^k$; $\Delta_{k+1} = \gamma_{inc}\Delta_k$;

    else

        $x^{k+1} = x^k$; $\Delta_{k+1} = \gamma_{dec}\Delta_k$;

    end

else

    $x^{k+1} = x^k$; $\Delta_{k+1} = \gamma_{dec}\Delta_k$;

end

$k = k + 1$ and go to **Start**;

Convergence

Under what assumptions will our algorithm converge almost surely to a first-order stationary point?

- Assumptions on f
- Assumptions on ϵ
- Assumptions on algorithmic constants

Convergence

Assumption

On some set $\Omega \subseteq \mathbb{R}^n$ containing all iterates visited by the algorithm,

- f *is Lipschitz continuous*
- ∇f *is Lipschitz continuous*
- f *has bounded level sets*

Assumption

The additive noise ϵ observed when computing $\bar{f}$ is independent and identically distributed with mean zero and bounded variance σ^2.

Convergence

Assumption

The constants $\alpha \in (0,1)$, $\gamma_{dec} \in (0,1)$, *and* $\gamma_{inc} > 1$ *satisfy*

$$\alpha \geq \max\left\{\frac{1}{2}, 1 - \frac{\frac{\gamma_{inc}-1}{\gamma_{inc}}}{4\left[\frac{\gamma_{inc}-1}{2\gamma_{inc}} + \frac{1-\gamma_{dec}}{\gamma_{dec}}\right]}\right\},$$

where

- α *is the lower bound on the probability of having a* κ*-fully linear model,*
- $\gamma_{dec} \in (0,1)$ *is the factor by which we decrease the trust region radius,*
- $\gamma_{inc} > 1$ *is the factor by which the trust radius is increased.*

Convergence

Assumption

The constants $\alpha \in (0,1)$, $\gamma_{dec} \in (0,1)$, *and* $\gamma_{inc} > 1$ *satisfy*

$$\alpha \geq \max\left\{\frac{1}{2}, 1 - \frac{\frac{\gamma_{inc}-1}{\gamma_{inc}}}{4\left[\frac{\gamma_{inc}-1}{2\gamma_{inc}} + \frac{1-\gamma_{dec}}{\gamma_{dec}}\right]}\right\},$$

where

- α *is the lower bound on the probability of having a* κ*-fully linear model,*
- $\gamma_{dec} \in (0,1)$ *is the factor by which we decrease the trust region radius,*
- $\gamma_{inc} > 1$ *is the factor by which the trust radius is increased.*

If $\gamma_{inc} = 2$ and $\gamma_{dec} = 0.5 \implies \alpha \geq 0.9$.
If $\gamma_{inc} = 2$ and $\gamma_{dec} = 0.9 \implies \alpha \geq 0.65$.

Proof Outline

Theorem

If the above assumptions are satisfied, our algorithm converges almost surely to a first-order stationary point of f.

Further Information and Current Work

Preprint on Optimization Online

"Stochastic Derivative-free Optimization using a Trust Region Framework"

Code

Just ask: `jmlarson@anl.gov`

Further Information and Current Work

Preprint on Optimization Online

"Stochastic Derivative-free Optimization using a Trust Region Framework"

Code

Just ask: `jmlarson@anl.gov`

- Generalizing results to ensure a practical algorithm converges.
  - For example, not requiring α-probabilistically κ-fully linear models every iteration.

Further Information and Current Work

Preprint on Optimization Online

"Stochastic Derivative-free Optimization using a Trust Region Framework"

Code

Just ask: `jmlarson@anl.gov`

- Generalizing results to ensure a practical algorithm converges.
  - For example, not requiring α-probabilistically κ-fully linear models every iteration.
- Smartly constructing α-probabilistically κ-fully linear models.

Problem Statement

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \mathbb{E}\left[\sum_{i=1}^{N} \bar{f}_i(x)\right] \\
\text{subject to} \quad & Ax \leq b \\
& x \in X
\end{aligned}
$$

- Each agent has objective $f_i(x)$ which can only be observed with additive noise $\bar{f}_i(x) = f_i(x) + \epsilon$
- Each f_i is convex
- ϵ has zero mean and finite variance
- X is an is a nonempty, closed, convex subset of $\mathbb{R}^n$

Problem Statement

$$\underset{x}{\text{minimize}} \quad \sum_i f_i(x)$$

$$\begin{aligned} &\underset{x}{\text{minimize}} && \sum_i f_i(x_i) \\ &\text{subject to} && x_i = x_j \qquad \forall (i,j) \in \mathcal{E} \end{aligned}$$

Problem Statement

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & Ax \leq b \\ & x \in X \end{aligned}$$

or

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

Previous Methods

Lagrangian dual decomposition methods (Nedić, Ozdaglar, Johansson, more)

- Challenge for using the dual when constructing models:

$$\begin{tikzcd}
\text{Primal Problem} \arrow[r, leftrightarrow, "\text{Global}"] \arrow[d, leftrightarrow, "\text{Local}"'] & \text{Dual Problem} \arrow[d, leftrightarrow, "??"] \\
\text{Model of Primal Problem} \arrow[r, leftrightarrow, "\text{Global}"] & \text{Dual of Model}
\end{tikzcd}$$

Previous Methods

Primal Methods using Consensus (Tsitsiklis, Bertsekas)

- Can be quite slow

Iterates have the form:

$$x_i^{k+1} = \sum_{j=1}^{N} w_{ij} x_j^k - a d_i^k$$

where a is a step size, d_i^k is an element of the subdifferential of f_i at x_i^k.

Previous Methods

Primal Methods using Consensus (Tsitsiklis, Bertsekas)

- Can be quite slow

Iterates have the form:

$$x_i^{k+1} = \sum_{j=1}^{N} w_{ij} x_j^k - a d_i^k$$

where a is a step size, d_i^k is an element of the subdifferential of f_i at x_i^k.

$$f(y^k) \leq f^* + aL^2 C_1 + \frac{NLBC_2}{k} \sum_{j=1}^{N} \left\| x_j^0 \right\| + \frac{N}{2ak} \left(\operatorname{dist}(y^0, X^*) + aL \right)^2$$

Nedić, Ozdaglar (2009)

Previous Methods

Alternating Direction Method of Multipliers (ADMM)

- Developed in the 1970s (Hestenes, Powell, Eckstein)
- Roots in the 1950s (Dantzig, Wolfe, Benders)
- Equivalent or similar to many other algorithms

Previous Methods

Alternating Direction Method of Multipliers (ADMM)

- Developed in the 1970s (Hestenes, Powell, Eckstein)
- Roots in the 1950s (Dantzig, Wolfe, Benders)
- Equivalent or similar to many other algorithms
  - Douglas-Rachford splitting
  - Spingarn's method of partial inverses
  - Dykstra's alternating projections
  - Proximal methods
  - Bregman iterative methods
  - More. . .

ADMM

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) + g(z) \\
\text{subject to} \quad & Ax + Bz = c
\end{aligned} \tag{1}
$$

has augmented Lagrangian

$$
L_\rho(x, z, \mu) = f(x) + g(z) + \mu^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2
$$

ADMM

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned} \tag{1}$$

has augmented Lagrangian

$$L_\rho(x, z, \mu) = f(x) + g(z) + \mu^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$

Algorithm 2: Traditional ADMM

Pick initial values z^0, μ^0, ρ;
for $k = 0, 1, \ldots$ **do**
  $x^{k+1} = \arg\min_x L_\rho(x, z^k, \mu^k)$;
  $z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k)$;
  $\mu^{k+1} = \mu^k + \rho \left(Ax^{k+1} + Bz^{k+1} - c\right)$;
end

Previous inexact ADMM methods

Algorithm 3: Deng, Yin (2013) Generalized ADMM

Pick $Q \succeq 0$ and symmetric P, z^0, μ^0, ρ;

for $k = 0, 1, \ldots$ **do**

$$x^{k+1} = \arg\min_x L_\rho(x, z^k, \mu^k) + \tfrac{1}{2}(x - x^k)P(x - x^k);$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k) + \tfrac{1}{2}(z - z^k)Q(z - z^k)\ ;$$

$$\mu^{k+1} = \mu^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right);$$

end

Previous inexact ADMM methods

Algorithm 3: Deng, Yin (2013) Generalized ADMM

Pick $Q \succeq 0$ and symmetric P, z^0, μ^0, ρ;

for $k = 0, 1, \ldots$ **do**

$$x^{k+1} = \arg\min_x L_\rho(x, z^k, \mu^k) + \tfrac{1}{2}(x - x^k)P(x - x^k);$$
$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k) + \tfrac{1}{2}(z - z^k)Q(z - z^k)\ ;$$
$$\mu^{k+1} = \mu^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right);$$

end

- Fixed matrices P and Q
- Still dealing with $\arg\min_x f$

Our approach

Algorithm 4: Our modification of ADMM

Pick initial values z^0, μ^0, ρ;

for $k = 0, 1, 2, \ldots$ **do**

$$x^{k+1} = \arg\min_x f(x^k) + \nabla_x f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla_x^2 f(x^k)(x - x^k) + (\mu^k)^T (Ax + Bz^k - c) + \frac{\rho}{2} \left\| Ax + Bz^k - c \right\|;$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k);$$

$$\mu^{k+1} = \mu^k + \rho \left(Ax^{k+1} + Bz^{k+1} - c \right);$$

end

Our approach

Algorithm 4: Our modification of ADMM

Pick initial values z^0, μ^0, ρ;

for $k = 0, 1, 2, \ldots$ **do**

$$x^{k+1} = \arg\min_x f(x^k) + \nabla_x f(x^k)^T(x - x^k) + \tfrac{1}{2}(x - x^k)^T \nabla_x^2 f(x^k)(x - x^k) + (\mu^k)^T(Ax + Bz^k - c) + \tfrac{\rho}{2}\left\|Ax + Bz^k - c\right\|;$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \mu^k);$$

$$\mu^{k+1} = \mu^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right);$$

end

Assumption

Assume f is convex and twice continuously differentiable in the region of interest so $\nabla^2 f(x^k)$ is well-defined.

Convergence

Assumption

There exists a saddle point to problem (1). In other words, there exists points x^*, z^*, μ^* satisfying

$$\begin{aligned} \nabla_z g(z^*) + B^T \mu^* &= 0 \\ \nabla_x f(x^*) + A^T \mu^* &= 0 \\ Ax^* + Bz^* &= c \end{aligned}$$

Define $\|x\|_A^2 = x^T Ax$ **and**

$$y^* = \begin{bmatrix} x^* \\ z^* \\ \mu^* \end{bmatrix}, y^k = \begin{bmatrix} x^k \\ z^k \\ \mu^k \end{bmatrix}, H_k = \begin{bmatrix} \nabla_x^2 f(x^k) + \rho A^T A & 0 & 0 \\ 0 & \rho I & 0 \\ 0 & 0 & \frac{1}{\rho} I \end{bmatrix}$$

Convergence

Lemma

Iterates generated by our algorithm satisfy

$$\left\|y^k - y^*\right\|_{H_k}^2 - \left\|y^{k+1} - y^*\right\|_{H_k}^2 \geq \left\|x^k - x^{k+1}\right\|^2_{(\nabla_x^2 f(x^k) + \rho A^T A - \frac{1}{\beta} A^T A)} + \left(\frac{1}{\rho} - \beta\right) \left\|\mu^k - \mu^{k+1}\right\|^2$$

for all $\beta > 0$*.*

Convergence

Lemma

Iterates generated by our algorithm satisfy

$$\left\|y^k - y^*\right\|_{H_k}^2 - \left\|y^{k+1} - y^*\right\|_{H_k}^2 \geq \left\|x^k - x^{k+1}\right\|^2_{(\nabla_x^2 f(x^k) + \rho A^T A - \frac{1}{\beta} A^T A)} + (\frac{1}{\rho} - \beta)\left\|\mu^k - \mu^{k+1}\right\|^2$$

for all $\beta > 0$*.*

- This shows y^k converges to y^* if $\nabla^2 f(x^k) \succ 0$.

Convergence

Lemma

Iterates generated by our algorithm satisfy

$$\left\|y^k - y^*\right\|_{H_k}^2 - \left\|y^{k+1} - y^*\right\|_{H_k}^2 \geq \left\|x^k - x^{k+1}\right\|^2_{(\nabla_x^2 f(x^k) + \rho A^T A - \frac{1}{\beta} A^T A)} + \left(\frac{1}{\rho} - \beta\right) \left\|\mu^k - \mu^{k+1}\right\|^2$$

for all $\beta > 0$.

- This shows y^k converges to y^* if $\nabla^2 f(x^k) \succ 0$.
- This shows y^k converges to some $\bar{y}$ if $\nabla^2 f(x^k) \succeq 0$.

Example: General ℓ_1 Regularized Loss Minimization

Consider the problem

$$\text{minimize } l(x) + \lambda \|x\|_1$$

where l is any convex loss function. In ADDM form, we can write this:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & l(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

where $g(z) = \|z\|_1$.

Example: General ℓ_1 Regularized Loss Minimization

Consider the problem

$$\text{minimize}\, l(x) + \lambda \|x\|_1$$

where l is any convex loss function. In ADDM form, we can write this:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & l(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

where $g(z) = \|z\|_1$.

- Instead of solving the x-update exactly, solving the quadratic approximation can be faster.

Results

$$\text{minimize} \sum \left(\log\left(-b_i(a_i^T x)\right)\right) + \lambda \|x\|_1$$

where a_i are rows in a feature matrix A and b is a response vector.

- Boyd's exact minimization (for a large problem) takes a total of 4928 iterations (summing over all agents)
- Solving only a single Newton step takes 1700 iterations

Concerns and Assumptions

Concerns

- Time varying network
- Asynchronous updates
- Delays in communication
- Nonconvex agent objectives

Concerns and Assumptions

Concerns

- Time varying network
- Asynchronous updates
- Delays in communication
- Nonconvex agent objectives

Assumptions

- Constant network
- Synchronized updates
- No delays in communication
- Convex agent objectives

Thanks

Questions?

jmlarson@anl.gov